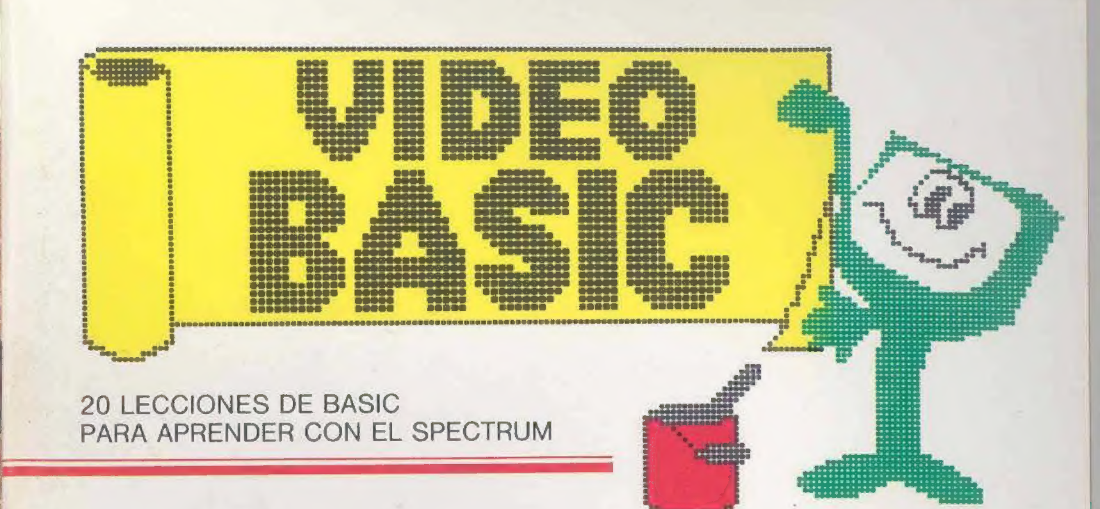


20 LECCIONES DE BASIC
PARA APRENDER CON EL SPECTRUM

INGELEK



JACKSON

La CPU del Spectrum
BUS, BIT Y BYTE
Sistema binario y hexadecimal
Técnicas de corrección
REM - GOTO - IF THEN - CLS
¿Verdadero o falso?
... La decisión del Spectrum
Videoejercicios
Videojuego N. 3

3

Spectrum
16K/48K/PLUS

VIDEO BASIC
Una publicación de
INGELEK JACKSON
Director editor por INGELEK:
Antonio M. Ferrer
Director editor por JACKSON HISPANIA:
Lorenzo Bertagnolio
Director de producción:
Vicente Robles
Autor: Softidea
Redacción software italiano:
Francesço Franceschini,
Stefano Cremonesi
Redacción software castellano:
Fernando López, Antonio Carvajal,
Alberto Caffaratto
Diseño gráfico:
Studio Nuovaidea
Ilustraciones:
Cinzia Ferrari, Silvano Scolari,
Equipo Galata
Ediciones INGELEK, S. A.
Dirección, redacción y administración,
números atrasados y suscripciones:
Avda. Alfonso XIII, 141
28016 Madrid. Tel. 2505820
Fotocomposición: Espacio y Punto, S. A.
Imprime: Rotacolor

ISBN del tomo 1: 84-85831-15-2
ISBN del fascículo: 84-85831-14-4
ISBN de la obra completa: 84-85831-13-6
Depósito Legal:
Plan general de la obra:
20 fascículos y 20 casetes, de aparición quincenal, coleccionables en 5 estuches.
Distribución en España:
COEDIS, S. A.
Valencia, 245. 08007 Barcelona.
INGELEK JACKSON garantiza la publicación de todos los fascículos y casetes que componen esta obra y el suministro de cualquier número atrasado o estuche mientras dure la publicación y hasta un año después de terminada.
El editor se reserva el derecho de modificar el precio de venta del fascículo, en el transcurso de la obra, si las circunstancias del mercado así lo exigen.

Mayo, 1985.
Impreso en España

INGELEK



JACKSON

SUMARIO

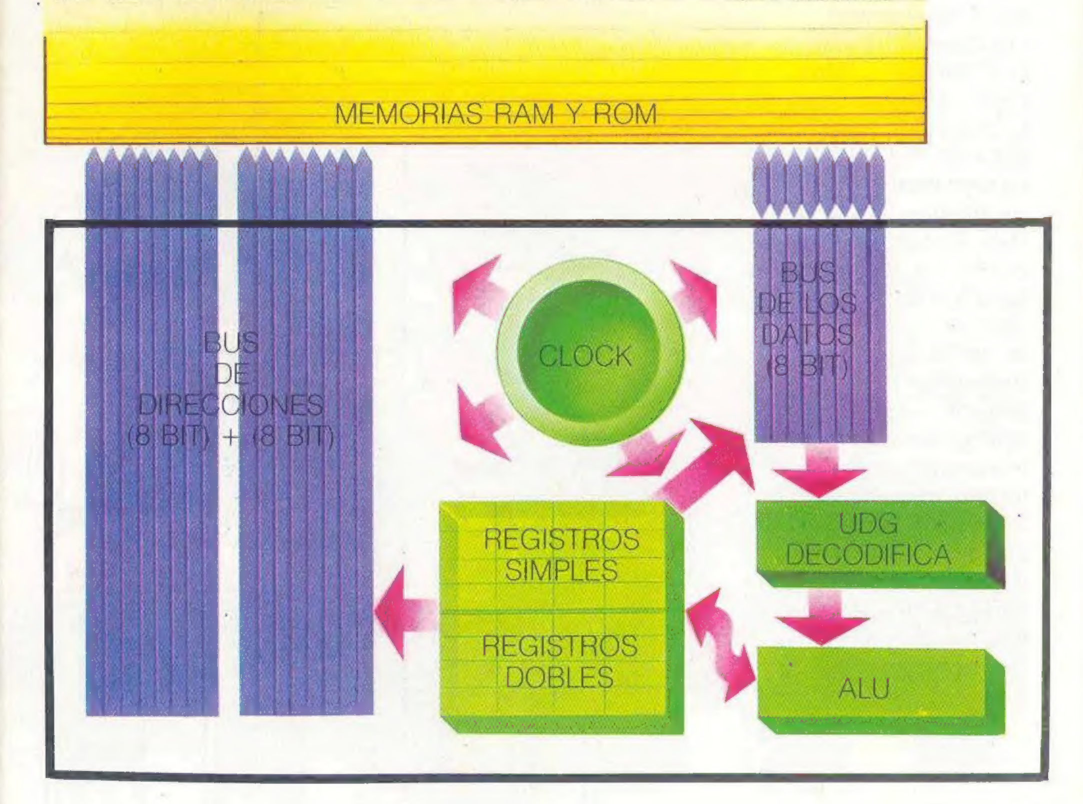
Introducción

Como verás, en la primera parte, hablaremos de la unidad fundamental de cualquier ordenador, la CPU, explicando cuál es su función y los códigos binario y hexadecimal.
Afrontaremos después el problema de cómo escribir bien los programas hablando del EDITOR, de cómo buscar y corregir los errores (DEBUG) e introduciremos las órdenes REM Y CLS.
Finalmente, tras haber aprendido el funcionamiento de la instrucción de salto condicional IF THEN GOTO, podremos hacer ejecutar a nuestro ordenador tareas de una cierta importancia, haciéndole operar con un ser «casi inteligente», gracias a su capacidad de decisión.

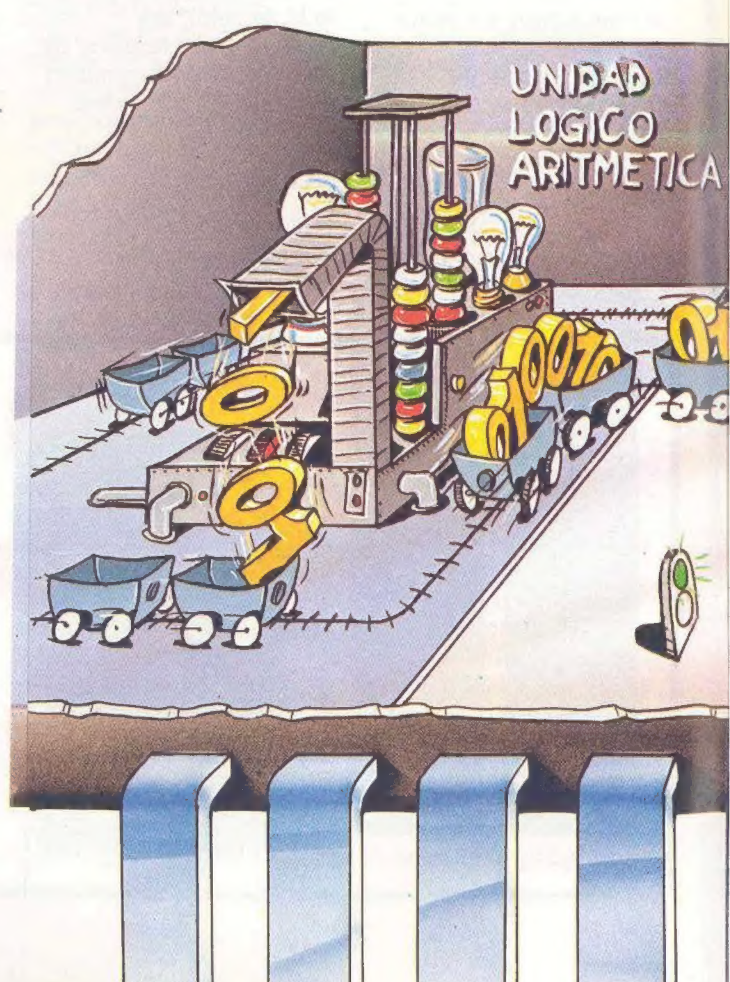
Un gran director: la CPU

Comenzamos por analizar las diversas unidades que componen el sistema de un ordenador, a partir del «cerebro» de nuestro sistema: la CPU. CPU significa «Central Processing Unit», que traducido al castellano significa «Unidad Central de Elaboración». Esta, fundamentalmente, se ocupa, de hacer ejecutar todas las instrucciones necesarias para el funcionamiento del ordenador y del programa.
Podemos ver como es de importante esta función por analogía con un concierto sinfónico.
El conjunto de la orquesta está representado por todo el conjunto de circuitos del ordenador, y el director de orquesta por la CPU. El director (la CPU) lee la partitura (el programa) e imparte órdenes a la orquesta, haciendo que los instrumentos (las diversas partes del ordenador) arranquen únicamente cuando se les requiera, y se paren si fuera necesario.
Después, el director, se ocupa de que todo esto ocurra armónicamente, o sea, de que el programa se desarrolle según esquemas precisos.
La CPU, «dirige» al ordenador.
A pesar de la onerosa carga que soporta sobre sus espaldas, la CPU, en su aspecto físico, no se diferencia mucho de otros componentes del interior de un ordenador, de modo que pasa casi desapercibida.
En el mercado existen diversos tipos de CPU, para satisfacer la mayor parte de las exigencias de quienes las usan: algunas son muy rápidas, otras más potentes, otras aún más flexibles, pero todas se parecen a arañas negras con muchas patas, y son, a simple vista, difícilmente diferenciables las unas de las otras.
Por suerte llevan una identificación: Z80A la CPU del Spectrum; 6510 la del C64; 6502, la del VIC 20
Los diversos modelos de CPU (como las interpretaciones de los directores de orquesta) no son equivalentes entre sí, y requieren instrucciones y circuitos distintos.
Una vez elegida la CPU sobre la que operar, se proyecta a su alrededor el ordenador y quedan así aproximadamente determinadas las prestaciones del ordenador.
De cualquier manera, todas las CPU, deben resolver un problema común: cómo ejecutar un conjunto de instrucciones.
Aclararemos mejor el problema. Todas las instrucciones tienen algo en común: suma, resta, enciende, apaga, pueden ser considerados como distintos casos de una misma tarea. Es como decir que estas instrucciones son consideradas iguales y analizadas del mismo modo. La diferencia está en la ejecución; en efecto, la compresión de la instrucción es algo bien distinto de su realización. En particular, una CPU considera de la misma manera todas la órdenes cuando «traduce» el programa a

sus instrucciones elementales y las pone en correspondencia con distintas acciones. Es como traducir al inglés desde el castellano: existen reglas generales y bien definidas de gramática y de sintaxis, y se puede hacer una traducción sin tener en cuenta el significado concreto de cada vocablo.
La traducción de estos últimos se realiza después aparte (la ejecución de la orden concreta).
En síntesis, cada máquina que «hace cuentas» (la CPU, por lo tanto) debe ser capaz de:
1) representar, registrar y manipular números;
2) identificar una instrucción;
3) ejecutarla;
4) elegir la sucesiva.
Para hacer esto se les requiere a algunas partes de la CPU (la ALU, el reloj, los registros, la memoria de solo lectura y la unidad de control) funciones bien definidas.
Pero antes de ver con detalle estas partes, es mejor ver como opera realmente una CPU.
Cuando enciendes el ordenador «das vida» a la CPU, que empieza leyendo un programa contenido en su interior. Este último, ordena a la CPU que ejecute el programa que empieza en la dirección 0 (dado que ella entiende el 0 como primer número) siguiendo, una tras otra todas las

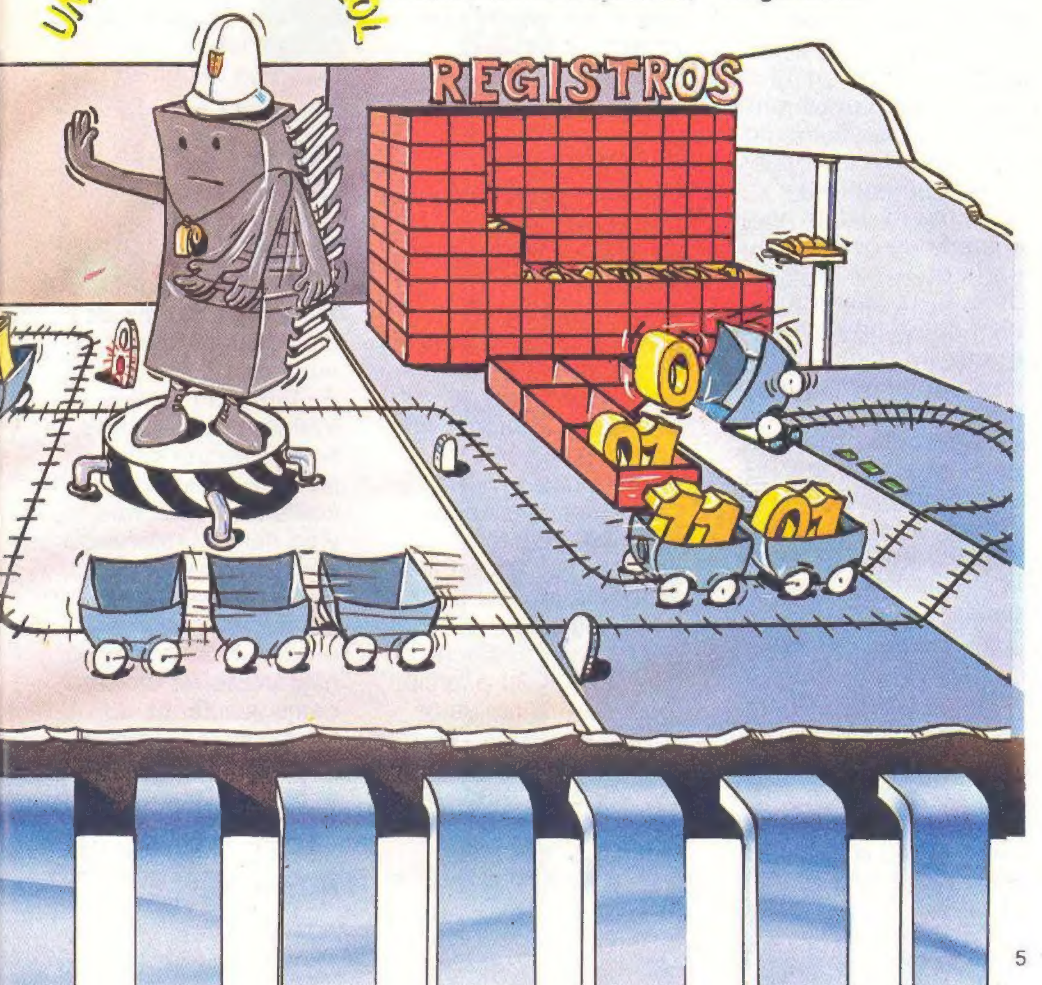


instrucciones paso a paso.
La CPU lee la instrucción en la dirección correspondiente, la compara con una lista, contenida en su interior, de las que puede ejecutar y encuentra la «acción» correspondiente a esa orden. En un último análisis cada instrucción se ejecuta a través de una serie de operaciones elementales, realizadas por los circuitos adecuados (internos de la CPU), los cuales, como en el ejemplo de la orquesta, son activados y desactivados en base al programa.
Para hacer esto es necesaria la presencia de las unidades antes citadas que son:
1) la ALU o Unidad Aritmético Lógica, que realiza todas las operaciones matemáticas y lógicas (como la comparación entre dos números);
2) la ROM, o memoria de sólo lectura (interna de la CPU), donde están contenidas permanentemente las informaciones de las instrucciones elementales propias de la CPU;
3) los registros, que son memorias temporales donde la CPU puede guardar números sin perderlos;
4) el reloj (clock) que marca el tiempo a todas las partes del microordenador, exactamente igual al metrónomo que marca el ritmo al interprete.
La ROM interior y el reloj (vistos en conjunto), se denominan comúnmente unidad de control, para subrayar su importancia en la gestión del «microsistema».



EL BUS

Un director de orquesta, para comunicarse con los músicos, emplea la batuta: ¿pero qué «batuta» emplea el ordenador?
En otras palabras, nosotros hasta ahora hemos dicho qué y cómo es capaz de actuar la CPU, pero no como esta última es capaz de poner a disposición de los demás circuitos, los datos que ella elabora.
En cada CPU existen estructuras para comunicar los datos y hacer conocer a otros circuitos las indicaciones de la CPU.
En la jerga, estos circuitos son llamados Bus, libremente traducible al castellano como «medio de comunicación» entre la CPU y el resto del ordenador.
En cada CPU existe más de uno, para poder afrontar todas las exigencias.



Exigencias que pueden ser:
a) la necesidad de comunicar qué posición de memoria está analizando la CPU (el llamado bus de direcciones);
b) la necesidad de dar a conocer el dato contenido en aquella posición (el bus de los datos);
c) la necesidad de conocer las informaciones referentes al manejo de los circuitos auxiliares (el bus de control).
Los Bus no deben entenderse como una parte de la circuitería de la CPU, sino más bien como un canal por donde se envían todas las informaciones referentes a un determinado problema. Es el Bus, y no la CPU, quien pone estos datos a disposición de todos los circuitos: la CPU se limita a colocarlos en el canal.

Bit

Ahora ya entrevemos la «lógica» de la CPU: ésta, aún conociendo únicamente dos estados (encendido o apagado), activa o desactiva un cierto número de circuitos.
La CPU, por lo tanto, adopta una lógica binaria (de dos estados).
Para abreviar, éstos se indican con 1 y 0.
Cada una de estas cifras se llama bit (de BInary digiT) y sobre ellas se fundamenta el álgebra binaria o de Boole, del nombre del matemático que la inventó.

Binario

Veamos ahora cómo se representan los números en lógica binaria.
Supongamos que deseamos saber cuánto vale en binario el número decimal 39.
Hay que establecer una premisa. Nuestro sistema de escritura de los números está en base 10, es decir emplea diez diferentes símbolos (de 0 a 9), y es un sistema posicional: la misma cifra, según su posición, toma un distinto valor (10^0, 10^1, 10^2,... 10^n).
Es decir cuando escribimos 39, damos por sobreentendida la siguiente relación:
$39 = 3 \times 10^1 + 9 \times 10^0 =$
$= 3 \times 10 + 9 \times 1$
Recordemos que cualquier número elevado a 0 es igual a 1.
En binario hemos visto que los estados posibles son 0 y 1, por lo tanto, decimos, que es un sistema en base dos (emplea dos símbolos).
Aquí podremos escribir, al igual que en los números decimales:

$00100111 = 0 \times 2^7 +$
$+ 0 \times 2^6 + 1 \times 2^5 +$
$+ 0 \times 2^4 + 0 \times 2^3 +$
$+ 1 \times 2^2 + 1 \times 2^1 +$
$+ 1 \times 2^0$

Volvamos, después de esta brevísima nota, a nuestro problema de pasar a binario el número 39.
Se toma el número a transformar a binario, se anota cuál es la máxima potencia de 2 contenida en el mismo, y se halla la diferencia entre ambos números, con el resultado se hace lo mismo y el procedimiento sigue hasta obtener un cero como resultado.
En nuestro caso el procedimiento es:
la máxima potencia de 2 contenida en 39 es 5:
$2^5 = 32$, $39 - 32 = 7$;





la máxima potencia contenida en 7 es 2: $2^2 = 4$, $7 - 4 = 3$; la máxima potencia contenida en 3 es 1: $2^1 = 2$, $3 - 2 = 1$; la máxima potencia contenida en 1 es 0: $2^0 = 1$, $1 - 1 = 0$; FIN. Donde no exista potencia de 2 se coloca un 0. Obtendremos por lo tanto: 100111. Otro método es el de dividir sucesivamente por dos el número decimal a convertir en binario, anotando, a partir de la derecha, los restos:

32768
16384
8192
4096
2048
1024
512
256
128
64
32
16
8
4
2
1

39 : 2 = 19 resto 1
19 : 2 = 9 resto 1
9 : 2 = 4 resto 1
4 : 2 = 2 resto 0
2 : 2 = 1 resto 0
1 : 2 = 0 resto 1

Por lo tanto 39 en binario corresponde a 100111.

El número 39 también podemos representarlo por:

$39 = 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 00100111$.

Hemos utilizado 8 cifras binarias en vez de 6 (00100111). Es como escribir 039. Esto, que no parece tener mucho sentido, en realidad es muy importante para un ordenador; ya que utiliza siempre números de igual longitud, por obvias exigencias prácticas. En el caso de tu Spectrum la longitud estándar es de 8 bits. Como el lenguaje del ordenador se basa en números, se ha pensado llamar a estas agrupaciones *palabra* o *byte*. El número más alto representable con un byte es 11111111 = 255. Pero, ¿y si quisiéramos representar un número

más alto?
Basta considerar dos byte consecutivos en la memoria como un único número para poder llegar a 65535.

Hexadecimal

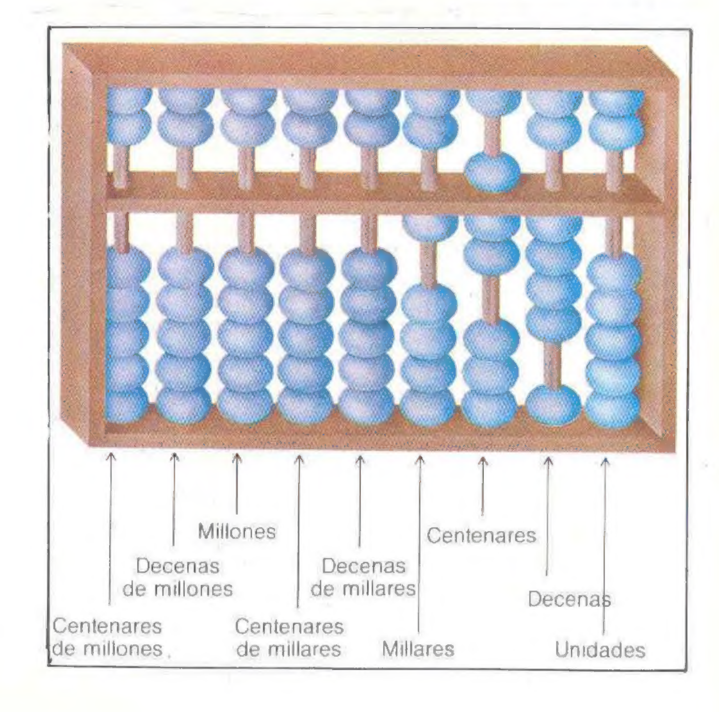
Por lo que acabamos de ver, para la representación de un número se emplean dos bytes consecutivos, considerando cada información como compuesta por 16 bits. Pero, dado que resulta poco práctico escribir 16 bits en fila para representar un número, se ha introducido el sistema de numeración en base 16 (hexadecimal), análogo a nuestro sistema decimal con la única diferencia de que además de los 10 símbolos usuales se emplean otros seis:

A=10 B=11 C=12
D=13 E=14 F=15

Por lo tanto, el número 20, se escribe 14 en hexadecimal, o bien 14 (decimal) se escribe simplemente E.
El hexadecimal es cómodo: es ciertamente más fácil leer D3 (de-3) que 11010011 (uno-uno-cero-uno-cero-cero-uno-uno).
Es necesario evitar leer los números hexadecimales como si fueran decimales. Por lo tanto, 20 (hex) se lee dos-cero y no 20, puesto que en efecto vale 32.

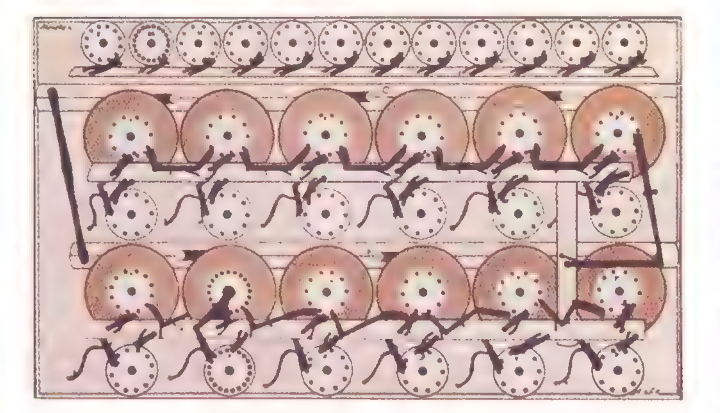
Apuntes históricos sobre el nacimiento de los ordenadores

Hemos analizado sucintamente el funcionamiento y el papel de la CPU en un ordenador.
Lo hemos hecho de forma que justifique sus singulares características (la aritmética binaria, los bus...) pero, no hemos explicado cómo se ha llegado a tal estructura. Y aún menos hemos aclarado el



El ábaco puede ser considerado como la primera máquina de calcular.

papel de la lógica. Históricamente, en efecto, la lógica matemática ha codificado los principios del cálculo matemático mucho antes de que el proceso científico y tecnológico hicieran posible la realización de ordenadores. La «idea» de ordenador ya era conocida mucho antes de que éstos en la común aceptación del término, fueran construidos.
Demos un alto atrás en el tiempo.
El hombre siempre ha tenido la necesidad de hacer cálculos, en forma exacta y rápida: al principio para preveer los fenómenos celestes,



La Pascalina hizo automática, por primera vez, la operación de llevar el resto.

después para determinar las rutas marítimas, luego para ejecutar las complejas operaciones ligadas al incremento del tráfico comercial.
Se piensa que la primera máquina de cálculo inventada por el hombres fue el ábaco: las famosas bolitas ensartadas en palos, aun hoy empleadas en algunas naciones. La idea de base es la de llevar más allá del número 10, las posibilidades de cálculo ofrecidas por los dedos.
Pero fue necesario esperar hasta los primeros años del siglo XVII, para llegar a la auténtica primera calculadora mecánica, obra del matemático frances Pascal. Aunque la Pascalina (éste fue su nombre), fuera capaz nada más que de realizar sumas y restas ha dado, a todos los efectos, el primer ejemplo de ordenador: dada una información de entrada, mediante complicadas rotaciones de los engranajes, se obtenía un resultado de salida.
Y ciertamente aunque se estuviera bien lejos del concepto de ordenador que tenemos hoy día, era la primera respuesta proporcionada por el hombre a la necesidad de cálculos automáticos.
Faltaba aun el concepto de secuencia de órdenes, y por lo tanto de control.
El paso adelante decisivo fue obra del lógico inglés Charles Babagge, que teorizó en 1834 la Máquina

Analitica.
En la máquina analitica, basada en un programa realizado sobre cinta de papel perforada (ideado por Jacquard para la programación de los telares), podemos ya reconocer las partes fundamentales de un ordenador moderno. Existia la unidad de cálculo, llamada fábrica o molino, donde eran realizados los cálculos, y existia memoria central, llamada almacén.
Lady Ada Lavelace, amiga de Babagge y colaboradora suya, afirmó al respecto: «podemos decir que la Máquina Analítica teje dibujos algebraicos, así como el telar de Jacquard teje flores y hojas». Por su actividad Lady Lavelace está considerada la primera programadora de la historia.
Pero Babagge fue derrotado por la inadecuada tecnologia de entonces y consiguió construir sólo en parte su Máquina Analítica.
De la cinta perforada se pasó con Holerith (el fundador de la IBM) a las fichas perforadas. Pero para tener el primer ordenador verdadero, hay que esperar hasta el año 1946, en el que nació el ENIAC: Calculador e Intregrador Numérico Electrónico.
Por primera vez,

Goldstine y Eckert con una unidad a vàlvulas del ENIAC.



si bien desde entonces sus historias ya viajarian juntas, la electrónica, con las válvulas, entraba en el ordenador.
Aun con millares de válvulas, que debían ser sustituidas cada seis horas so pena de quemarse (se calentaban hasta tal punto que eran necesarias plantas de refrigeración), no se conseguían elevadas velocidades de cálculo. (Piensa que tu Spectrum es mucho más potente, veloz y versátil que el ENIAC, aunque este último ocupara la superficie entera de un gimnasio).
En 1948 J. Bardeen, W. Brattain y W. Schockley anuncian el descubrimiento de un componente fundamental: el transistor.
El transistor cumple todas las funciones de una válvula, pero con numerosas ventajas: ocupa menos espacio, disipa poco calor, es muy rápido y casi eterno.
Era inevitable que fuera aplicado a los ordenadores, dando origen a la llamada «nueva generación».
Después, gracias a su menor costo, los ordenadores tuvieron una gran difusión y empezó entonces la «batalla» de las prestaciones, una incruenta lucha para la mejora de las caracteristicas.
Del transistor en adelante, la electrónica concentró todos sus esfuerzos en la integración: miniaturización y compactación de los componentes básicos (entre ellos el citado transistor) en un espacio siempre más reducido.
Reducir los componentes a dimensiones visibles únicamente al microscopio no es cuestión de poco y el camino desde 1958 (año en el que fue construido el primer circuito integrado) hasta hoy está jalonado por sucesivas mejoras:

- años 60, SSI (Pequeña Escala de Integración), 12 puertas lógicas por circuito integrado;
- fin de los años 60, MSI (Media Escala de Integración), 100 puertas lógicas por circuito integrado;
- años 70, LSI (Gran Escala de Integración), 1.000 puertas lógicas por circuito integrado;
- Fin de los años 70, VLSI (Grandísima Escala de Integración), más allá de 50.000 puertas lógicas por circuito integrado.

Hoy en un cm^2 de superficie se pueden colocar cien mil transistores.
La historia del ordenador y de la electrónica está, de cualquier forma bien lejos de agotarse; en efecto, se ha calculado que el progreso en el campo de la integración no ha alcanzado aún su máximo nivel, y hay que esperar una nueva «revolución» en este campo.
Tu Spectrum es hijo de esta tecnología, y sin esta última no sería capaz, en tan poco espacio y a bajo precio, de realizar todas las funciones que aprenderás a conocer.

LENGUAJE

REM

La REM (abreviatura del término inglés Remark, comentario) es una instrucción que... ¡parece que no sirve para nada!
Cuando tu Spectrum encuentra una REM, efectivamente la ignora por completo y pasa a ejecutar la línea siguiente del programa.
En realidad REM es una instrucción importante: te permite insertar comentarios, títulos y descripciones, sin que éstos alteren el desarrollo normal del programa.
Su argumento puede ser una secuencia cualquiera de caracteres, también símbolos gráficos y matemáticos y hasta una línea vacía.
Puedes insertar una REM en cualquier parte del programa, o al final de cada línea de instrucciones, basta con respetar las reglas de sintaxis de tu Spectrum.
Después de la REM no puedes insertar ninguna instrucción o comando, porque sería considerado todo como un comentario.



Ejemplos

```
10 REM "GEOMETRIA 1"
```

Esta línea da el título "GEOMETRIA 1" a tu programa.

Cuando al cabo de un tiempo, lo vuelvas a usar, no tendrás que analizar las instrucciones para entender si el programa en cuestión es una contabilidad doméstica o un archivo de discos. Tienes la respuesta en la primera línea del programa.

```
40...
50...
60...
70 INPUT B
80 REM CALCULO DE LA MEDIA
```

Te comunica que las líneas a continuación de la REM sirven para el cálculo de la media.

```
90 LET M = (B + C + F + G)/4
100...
110...
120...
```

Cuando te sea necesario calcular la media de varios números para algún otro programa sabrás dónde encontrar las instrucciones necesarias sin necesidad de reescribirlas.
O bien: si en el lugar de la media necesitas obtener la raíz cuadrada de esos números, podrás simplemente sustituir las líneas siguientes a la REM, sin tener, por esto, que volver a analizar y modificar todo el programa.

```
30 REM "GEOMETRIA": PRINT GEOMETRIA
```

Tu Spectrum interpreta todo el argumento de REM ("GEOMETRIA" : PRINT GEOMETRIA) como un comentario, y no ejecutará nunca la

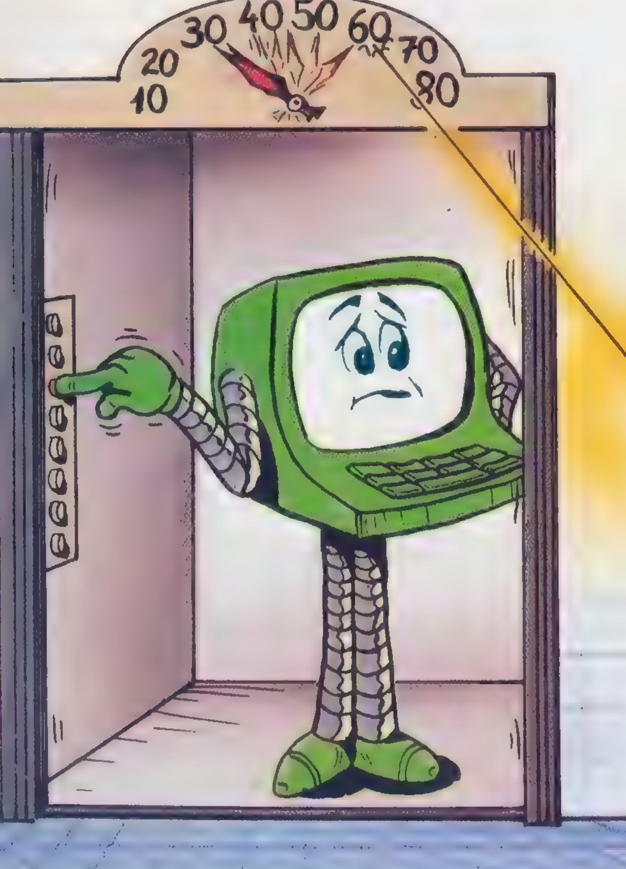
instrucción contenida (PRINT).

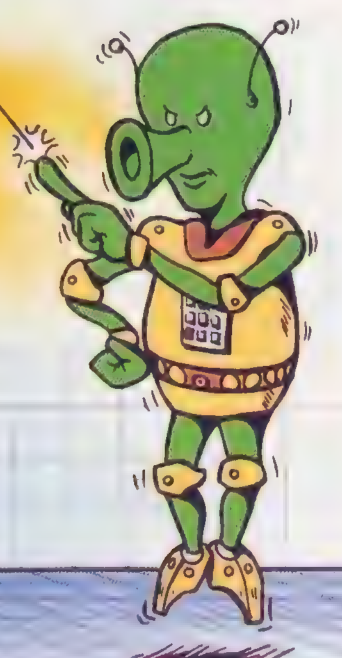
Sintaxis de la instrucción

REM cualquier secuencia de caracteres

GOTO

Al hablar del algoritmo, habrás visto que éste contine todas las informaciones necesarias para llevar a término una elaboración.
Pero el algoritmo, no debe forzosamente desarrollarse secuencialmente (una instrucción detrás de la otra), sino que puede proceder a «saltos», para poder modificar según sea necesario el orden de las instrucciones a ejecutar.





LENGUAJE

El BASIC posee una instrucción que realiza esta tarea: GOTO.
La instrucción GOTO (salta a...) siempre es seguida por un número en forma explícita (o bajo forma de variable o expresión). En la práctica, cuando tu Spectrum encuentra una GOTO, en vez de continuar con la instrucción siguiente salta a aquélla indicada, continuando desde ésta como si nada hubiera ocurrido.
(¡Pero cuidado!, porque si esta última no existiera, el intérprete BASIC te enviará el mensaje de error <UNDEF STATEMENT>, precisamente para advertirte de que esa línea no existe).
En resumen, con GOTO te es posible modificar el orden de ejecución en base a tus necesidades.
Un ejemplo claro es el cálculo de la tabla de los cuadrados.

```
10  REM CALCULA LOS CUADRADOS DE LOS NUMEROS
20  REM A PARTIR DEL 1
30  REM HASTA QUE NO SE LE PARE
40  LET NUMERO = 0
50  REM AÑADE 1 EN CADA PASO
60  LET NUMERO = NUMERO + 1
70  REM IMPRIME EL NUMERO Y SU CUADRADO
80  PRINT NUMERO, NUMERO↑2
90  REM REPITE INCREMENTANDO
100 GOTO 50
```

GOTO es llamada también, instrucción de salto incondicional, puesto que ordena al Spectrum que salte en cualquier caso a la línea indicada
Por ejemplo, observa el programa que sigue:

```
10 GOTO 40
20 ...
30 ...
40 REM
```

El programa empieza siempre en la línea 40, saltándose las líneas 20 y 30.

Esta es su aplicación más común. De cualquier manera, existen algunos casos especiales.
Uno de éstos es:

```
90 GOTO 90
```

o, en general:

```
90 instrucción cualquiera: GOTO 90
```

La primera se emplea en casos particulares para interrumpir el flujo del programa sin que éste realmente se pare: el calculador efectúa un infinito número de saltos a la línea 90.
La segunda es equivalente a 90 GOTO 90, con la única diferencia de que tu Spectrum ejecuta un número infinito de veces las órdenes especificadas antes de la instrucción GOTO.
De cualquier manera, ambas sirven para crear programas que no se paran nunca, si no es por una orden exterior.
Prueba con este programa, para convencerte:

```
10 REM PROGRAMA QUE SE INTERRUMPE
20 REM SIN PARARSE
30 PRINT "ESTOY EN LA LINEA 30"
40 PRINT "ESTOY EN LA LINEA 40" : GOTO 40
```

Ejemplos

```
30...
40...
50...
60  GOTO 100 + (ANGULO * 4)
```

Como ya se ha dicho, el argumento de GOTO puede ser una expresión cualquiera, que dé un número como resultado. En este caso, si ANGULO = 100, cuando el programa llegue a la línea 60 saltará a la
100 + (100 * 4) = 500.

```
30...
40...
50...
60 GOTO A
```

El argumento de goto puede ser también una variable numérica. Cuando A vale 100, GOTO hará saltar la ejecución del programa a la línea 100. Si ésta no existiera, a la línea inmediatamente sucesiva.

```
70 LET A = 100
80 GOTO A
90 REM
105 PRINT A
```

Puesto que falta la línea 100, la ejecución sigue desde aquélla inmediatamente sucesiva: en el ejemplo la línea 105.

```
10 REM ¿ES UN CUADRADO O UN ROMBO?
20 INPUT "NUMERO DE LADOS IGUALES" ; L
30 INPUT "NUMERO DE ANGULOS IGUALES" ; A
40 GOTO 20
50 IF L = A THEN PRINT "ES UN CUADRADO"
60 PRINT "NO ES UN CUADRADO"
```

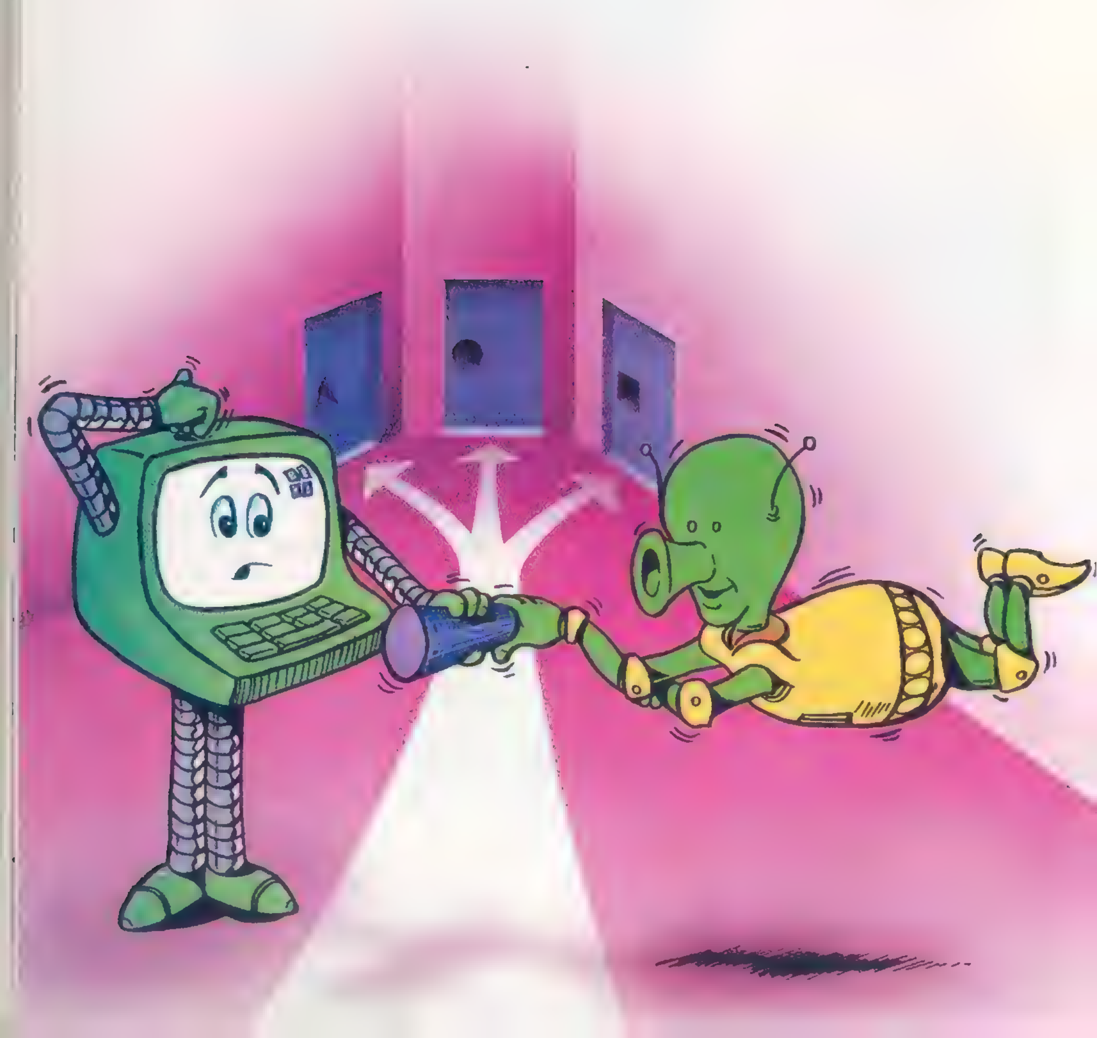
El argumento de la instrucción GOTO debe ser cuidadosamente escogido: un valor equivocado puede tener consecuencias catastróficas sobre tu programa. Si consideras este programa, verás que tu Spectrum nunca llegará a las instrucciones de las líneas 50 y 60.

Sintaxis de la instrucción

	número
GOTO	expresión numérica
	variable

IF THEN

Como ya tedremos ocasión de puntualizar más adelante (cuando continuemos el tema del Diagrama de flujo), una parte fundamental de la programación se ocupa de las decisiones. Es gracias a ellas como conseguimos adaptar una solución general (el programa) a la situación particular (el problema a resolver). Sin algo que le permita al programa cambiar su curso, en base al contexto, muchas aplicaciones del ordenador no serían posibles: deberíamos escribir un



programa rigidamente secuencial.
Por otra parte, en nuestra vida, las decisiones desempeñan un papel fundamental: ocurre que nosotros infravaloramos su importancia porque estamos desde siempre, acostumbrados a tomarlas.
El ordenador, sigue siendo importante subrayarlo, es un ejecutor formidable al que hay que decirle todo lo que tiene que hacer (hasta las cosas más obvias), puesto que carece de raciocinio y de iniciativa propia.
Por lo tanto, la decisión es una de las piedras angulares de la programación.
¿Qué es lo que puede decidir un ordenador? Hablando de la CPU, hemos dicho que el ordenador analiza únicamente números, y sabe decir si dos números son iguales o distintos.
Para reducir cualquier decisión a una forma «comprensible» para el ordenador es por lo tanto necesario, reducir el problema a números.
Si número 1 = número 2, entonces escribirá «los dos números son iguales».
Esta es la forma computable, y, generalizando, en BASIC se escribe:

```
IF CONDICION LOGICA THEN INSTRUCCION
```

Supón que hayas asignado a la variable X un número secreto que pretendes hacerle acertar a un amigo, insertando en la variable T el intento de resolución. Lo que necesitas es una instrucción que le ordene a tu Spectrum que señale cuando el número secreto haya sido acertado, o sea, cuando la variable T sea igual a la variable X.

```
SI T = X ENTONCES IMPRIME "OK"
```

Bien, no deberías tener muchos problemas para escribir una instrucción de este estilo, pues ya lo has hecho antes...

Es suficiente con que traduzcas al inglés la instrucción, tal como la has pensado, y tendrás:

```
IF T = X THEN PRINT
"OK"
```

Resulta evidente la importancia de IF... THEN...: sirve para que el ordenador efectúe auténticas elecciones, en base a las cuales el programa pueda ejecutar, en cada caso, la función más adecuada para una circunstancia particular. Es un poco como dotar a tu Spectrum con la capacidad de discernir, casi un «cerebro».

Naturalmente, en lugar de PRINT, puedes escribir cualquier otra instrucción, así como puedes igualmente, indicar comparaciones, (mayor o menor), de valores númericos o alfanuméricos.
En el caso de que la condición no sea cierta, es completamente ignorada la continuación de la línea, y el control pasa a la siguiente.
Recuerda pues: toda la parte que sigue al THEN está sujeta a comparación.
Esta posibilidad particular ofrece notabilísimas ventajas: te permite escribir grupos completos de instrucciones controlados por un solo IF, ahorrando GOTO superfluas y obteniendo programas compactos, veloces y muy legibles.
Aún así, te ocurrirá frecuentemente que encuentres IF... THEN GOTO, puesto que la combinación de estas dos instrucciones genera auténticas ramificaciones del programa, conceptualmente muy similares a cambios de vía ferroviarios, los cuales, según su colocación, dirigen a uno u otro recorrido.
Veamos una aplicación simple pero útil de IF THEN GOTO, retomando uno de los ejemplos ya vistos (90 GOTO 90).
Aportémosle una pequeña modificación:

```
80 PAUSA = 0
90 PAUSA = PAUSA + 1 : IF PAUSA < 1000 THEN GOTO 90
```

Obtenemos un «timer» (temporizador): tu ordenador cuenta hasta 1000 (en este caso) antes de seguir con la instrucción siguiente.
Este puede ser un recurso válido cuando tienes necesidad de retrasar la ejecución de algunas instrucciones.

En el caso de que debas comparar valores alfanuméricos, el ordenador asignará a cada letra del alfabeto, símbolo gráfico, etc., un determinado número, según un standard definido. Este código es proporcional a la posición alfabética del carácter: por ejemplo, A corresponde a 65, B a 66, C a 67... y así sucesivamente. Ahora la comparación es sencillísima, reduciéndose al cotejo de los códigos que componen las dos cadenas. Después de una instrucción IF THEN puedes omitir la palabra reservada GOTO, indicando únicamente el número de línea a la que saltar. El resultado será el mismo, un salto a la línea indicada.

Ejemplos

```
10 PRINT "¿CUANTO DINERO TIENES EN EL
   BOLSILLO?"
20 INPUT DINERO
30 IF DINERO ≥ 3000 THEN PRINT
   "¡ERES RICO!"
40 IF DINERO < 3000 THEN PRINT
   "¡NO TIENES MUCHO!"
50 IF DINERO = 0 THEN PRINT "¡NO
   TIENES NADA!"
60 $TOP
```

Analicemos el programa adjunto: si una de las tres expresiones es verdadera el programa imprime el comentario apropiado. Si tuvieras 500 Ptas., el resultado sería: "¡NO TIENES MUCHO!", puesto que la segunda instrucción IF es verdadera.

```
70 IF TV = 30 THEN LET RADIO = 24
```

Si la variable TV vale 30, entonces la variable RADIO toma el valor 24.

```
70 IF NUMERO = 88 THEN IF OTRO NUMERO
   = 66 THEN GOTO 90
```

La orden IF THEN tiene algunas características interesantes, como la posibilidad de concatenar secuencialmente más de una instrucción. Este procedimiento se llama anidamiento, puesto que reagrupa (anida) distintas instrucciones como parte de una única instrucción. En el ejemplo, si el número es

88, el ordenador comprueba que el otro sea 66 antes de pasar a la línea 90.

Virtualmente no existe límite en el anidamiento de IF THEN; sin embargo, conviene no abusar para evitar problemas de corrección (DEBUGGING).

```
IF RUEDAS = 4 AND VEHICULO = 1 THEN PRINT
"ES UN COCHE"
```

IF comprueba que una expresión lógica sea verdadera: puedes entonces insertar operadores lógicos que te permitan ampliar el control de la IF a más de una expresión, como se ha hecho en el ejemplo. La expresión RUEDAS = 4 AND VEHICULO = 1, es verdadera únicamente cuando ambas igualdades se verifican, y sólo en este caso el ordenador imprime el comentario apropiado.

```
40 IF DINERO <100 THEN PRINT "¡NO TIENES
MUCHO!"
```

Debes poner atención en la elección del argumento de IF. Si en el ejemplo "¿CUANTO DINERO TIENES EN EL BOLSILLO?", pones en la línea 40 y dices que tienes 2500, el ordenador, como podrás comprobar, ¡no te contestará nada!

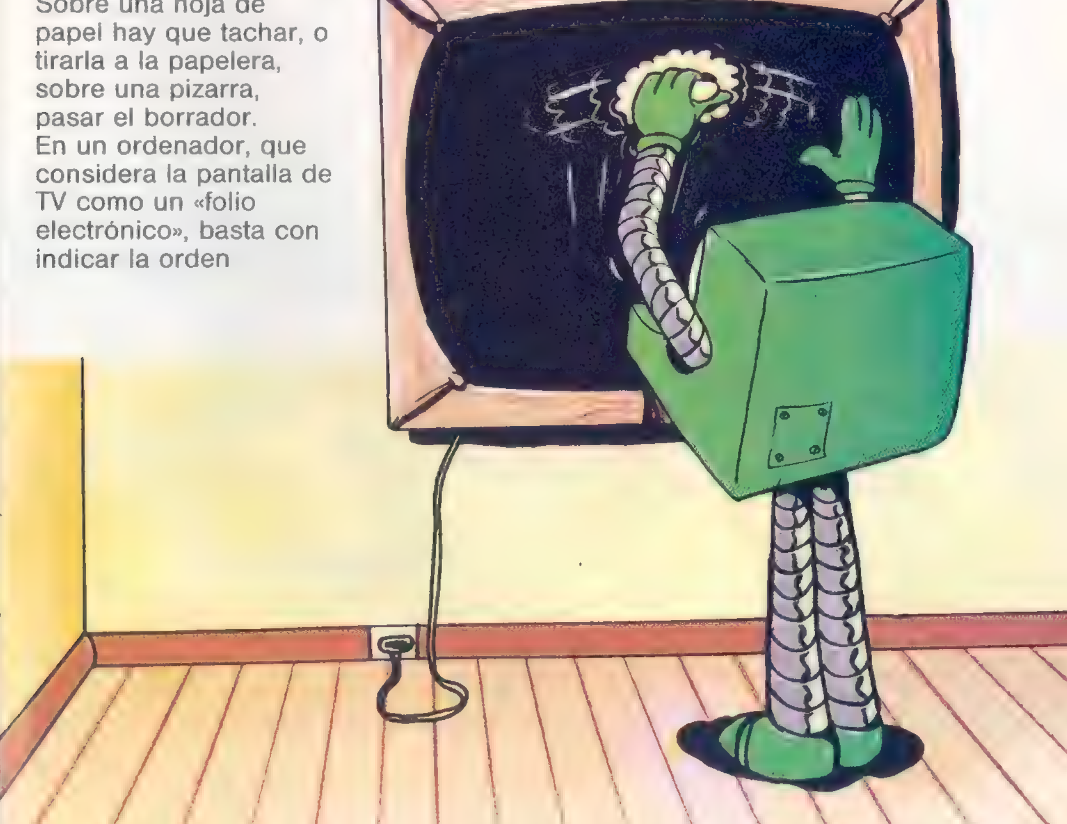
Sintaxis de la instrucción

IF expresión lógica THEN instrucción ejecutable

LENGUAJE

CLS

Con frecuencia nos ocurre que tengamos que escribir lo que decimos, sea para recordar alguna cosa, sea porque necesitemos enseñar a otros lo hecho.
Con la misma frecuencia nos encontramos en la necesidad de borrar lo escrito, porque ya no sirve o porque está equivocado.
Sobre una hoja de papel hay que tachar, o tirarla a la papelera, sobre una pizarra, pasar el borrador.
En un ordenador, que considera la pantalla de TV como un «folio electrónico», basta con indicar la orden apropiada.
En nuestro caso CLS, del inglés Clear Screen, limpia la pantalla.
Con solo estas tres letras podrás vaciar a tu gusto la pantalla de todo lo presente.
Tu Spectrum considera la pantalla como un conjunto de muchos pequeños puntos; no te será difícil verlos acercándote a la pantalla cuando hayas escrito algo.
Cuando el ordenador encuentra la instrucción CLS «pinta» estos puntos con el mismo color del fondo (en la práctica los borra) y coloca el cursor en la esquina superior izquierda.
Si en un programa necesitas visualizar muchas cosas, no debes ahorrar los CLS: tus ojos lo agradecerán.



Como escribir programas: técnicas de claridad y corrección

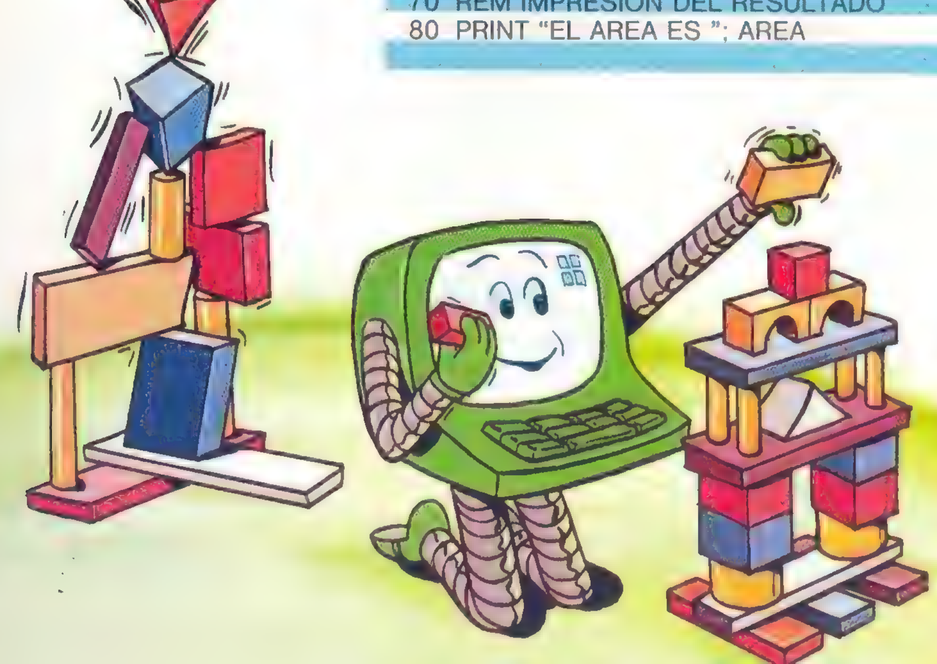
Ya hemos visto algunos programas sencillos, ¿pero cuáles son los requisitos necesarios para escribir un buen programa? A un programa «ideal» normalmente le pedimos que ejecute todas las operaciones necesarias en el menor tiempo posible, que sea fácilmente modificable y corregible y que ocupe la menor cantidad posible de memoria. Pero no podemos, excepto en casos particulares, variar en mucho la velocidad de cálculo, y de hecho, el único parámetro sobre el que podemos influir significativamente, sigue siendo la corregibilidad del listado. En efecto, la tarea más ingrata en la redacción de un programa normalmente es aquélla de encontrar y corregir los errores cometidos, operación que en la jerga, llamamos debugging, que traducido literalmente, significa «expurgar».
El problema es más grave cuando el lenguaje ofrece muchas (demasiadas) posibilidades de aproximación al problema mismo, como ocurre en el BASIC. Muchas veces, no sabiendo en qué zona del programa está escondido el error, se debe analizar el listado



desde el principio hasta el final, con gran gasto de tiempo y trabajo, precisamente porque en él están contenidas desordenadamente, variables e instrucciones.
El primer paso para una mejor redacción de los programas es darle un orden a su interior: crear una estructura.
Así, es buena norma insertar en primer lugar las variables, de forma que estén inmediatamente disponibles para su comprobación y/o corrección. Haciendo esto, entre otras cosas, aumentaremos ligeramente la velocidad de ejecución del programa, dado que en su desarrollo el ordenador únicamente leerá instrucciones «activas», que consisten en operaciones que llevan a la elaboración de los datos en examen, memorizándose todas las variables al principio.
Otra dificultad que se encuentra durante la redacción de un programa es su abstracción. Casi todos los lenguajes existentes están ligados, en medida más o menos acentuada, al modo en el que la máquina elabora las informaciones que le proporcionamos. Y esta «lógica» está muy alejada de nuestra manera de actuar y de pensar, lo que nos obliga a «traducir» nuestra lógica a la del ordenador, trabajo no del todo fácil.
Por lo tanto, es aconsejable adoptar algunas precauciones para hacer menos abstracto y formal un programa, y por lo tanto más cercano a quien lo utilice.
Una primera técnica consiste en emplear, en la definición de una variable, nombres compuestos que recuerden fácilmente cuál será su uso. Por ejemplo, para programar el Teorema de Pitágoras, podrías emplear como variables CATETO1, CATETO2, HIPOTENUSA; a, b, c también hubieran sido nombres correctos.
Aunque no todos los ordenadores prevean este tipo de variables, tu Spectrum, afortunadamente, es capaz de reconocer variables compuestas.
Otro tipo de estrategia a adoptar, es la de insertar en los puntos clave de los programas, comentarios, e indicaciones sobre lo que el bloque de instrucciones en cuestión está destinado a realizar. De esta manera, aún habiendo pasado tiempo, te será suficiente con una ojeada para reconocer las distintas partes de un programa, ganando en claridad y posibilidades de modificación.
En BASIC esta posibilidad te la proporciona el comando REM, que permite escribir un texto cualquiera sin que el programa lo tenga en cuenta.
Otro requisito esencial para la claridad de un programa es su linealidad, es decir, que el programa proceda secuencialmente, una línea detrás de la otra, con el menor número posible de «saltos». En el lenguaje BASIC la

instrucción «salto a una línea» es GOTO y deberás intentar hacer de ella el menor uso posible, para facilitar al máximo el debugging. Si el programa por tí escrito e ideado, prevee distintas posibilidades de cálculo, en base a las necesidades de quien lo use, será conveniente dividir enseguida sus diversas posibilidades, separándolas en módulos diferentes en el interior del listado. Y siempre, por exigencias de claridad, es conveniente escribir líneas de programa no demasiado largas, para evitar complicaciones de lectura. Resumiendo todos los conceptos expuestos, podemos ahora escribir un programa BASIC que se acerque más a nuestras exigencias. Retomemos, como ejemplo, el programa de la superficie de un triángulo. Empezaremos poniendo como primera línea del listado una REM que nos explique la finalidad del programa. Después seguiremos insertando otras REM que nos recuerden qué hacen los distintos módulos del programa. En este punto obtendremos:

```
10 REM LA SUPERFICIE DE UN TRIANGULO
20 REM INTRODUCCION DE LOS DATOS
30 INPUT "BASE = "; BASE
40 INPUT "ALTURA = "; ALTURA
50 REM CALCULO DE LA SUPERFICIE
60 LET AREA = BASE * ALTURA/2
70 REM IMPRESION DEL RESULTADO
80 PRINT "EL AREA ES "; AREA
```



Como habrás notado las variables empleadas usan nombres extendidos que declaran explícitamente su contenido.
Una lectura del programa podría ser:

10 ENCABEZAMIENTO
20 : 40 MODULO DE ENTRADA DE DATOS
50 ÷ 60 MODULO DE ELABORACION
70 : 80 MODULO DE SALIDA DE DATOS

Las primeras veces te parecerá artificioso programar por estructuras, pero después de poco tiempo apreciarás sus ventajas, la claridad de los listados, el ahorro de tiempo y cansancio durante y después de la programación.
El conjunto de reglas que permiten cambiar un programa añadiendo y/o corrigiendo sus partes, se llama EDITOR (en castellano, algo parecido a «corrector» o «supervisor»).
Cada ordenador (también tu Spectrum, tiene un programa editor que permite efectuar estas modificaciones: podrás informarte de sus posibilidades leyendo el manual de instrucciones. Es bueno adueñarse, en seguida de estos instrumentos de corrección, para ahorrar esfuerzos inútiles durante y después de la corrección.

Las decisiones

Ahora tienes que aprender a comunicarte con el ordenador, aunque sólo sea para que realice algunas simples operaciones aritméticas.
Pero su valor y sus capacidades quedarían limitadas si se redujeran exclusivamente a reproducir modelos ya preconstituidos. De hecho, después podremos hablar de «inteligencias artificiales» precisamente porque los ordenadores «deciden» que hacer (con respecto a la realización de las tareas que les encomendamos a través de los programas) en base a criterios lógicos y de valor, ¡esto sí!, anteriormente preconstituidos.
En BASIC, las decisiones se ejecutan empleando la instrucción IF... THEN, que le permite a tu Spectrum

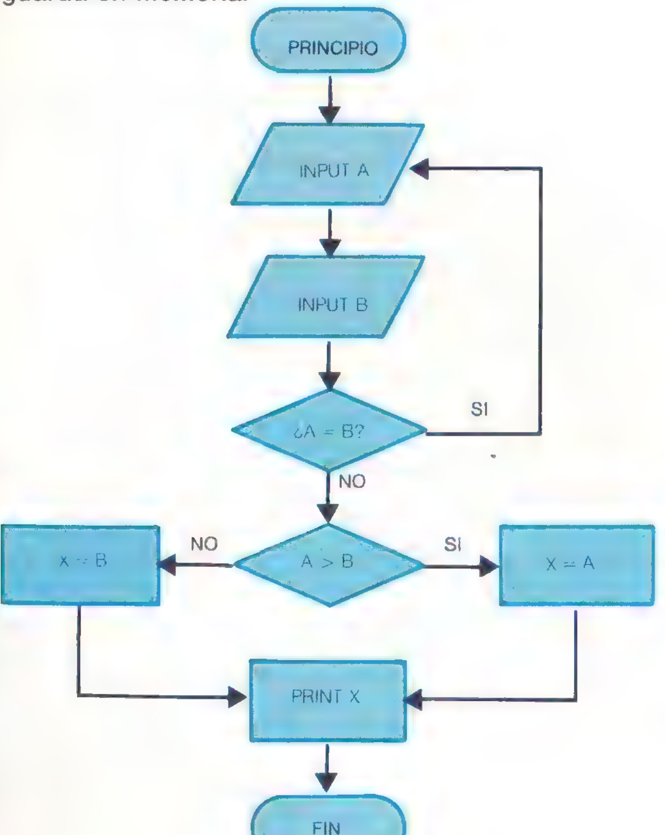
proporcionarte el resultado de un TEST y luego ejecutar una instrucción en lugar de otra.
Presentamos ahora un primer y sencillo problema para ver cómo puedes implementar un programa en el cual el ordenador realiza una «elección» en base a los «conocimientos» que guarda en memoria.

Objetivo

Queremos, por ejemplo, hacer determinar a nuestro ordenador cuál entre dos números cualquiera, que denominamos A y B, es el mayor.
Deseamos por lo tanto, que el ordenador haga tomar a una incógnita, que llamamos X, el valor más alto.

Procedimiento

Para resolver este problema, primero hay que cotejar A y B. Si A>B, entonces se pone el valor A en X, de otro modo en la X se pone el valor de B. Este modo de proceder en la resolución puede ser representado gracias al diagrama de flujo, que contiene



Pide el primer número

Pide el segundo número

Si son iguales vuelve atrás, si no...

Comprueba cuál de los dos es el mayor

...y lo imprime

dos «rombos» en los cuales se encuentra el cotejo entre A y B, y dos «rectángulos», que corresponden a las «asignaciones».
El listado siguiente corresponde a su secuencia en BASIC.

```
10 REM EL NUMERO MAYOR ENTRE 2
20 INPUT "PRIMER NUMERO = "; A
30 INPUT "SEGUNDO NUMERO = "; B
40 IF A = B THEN GOTO 20
50 IF A>B THEN LET X = A
60 IF A<B THEN LET X = B
70 PRINT "EL NUMERO MAYOR ES"; X
```

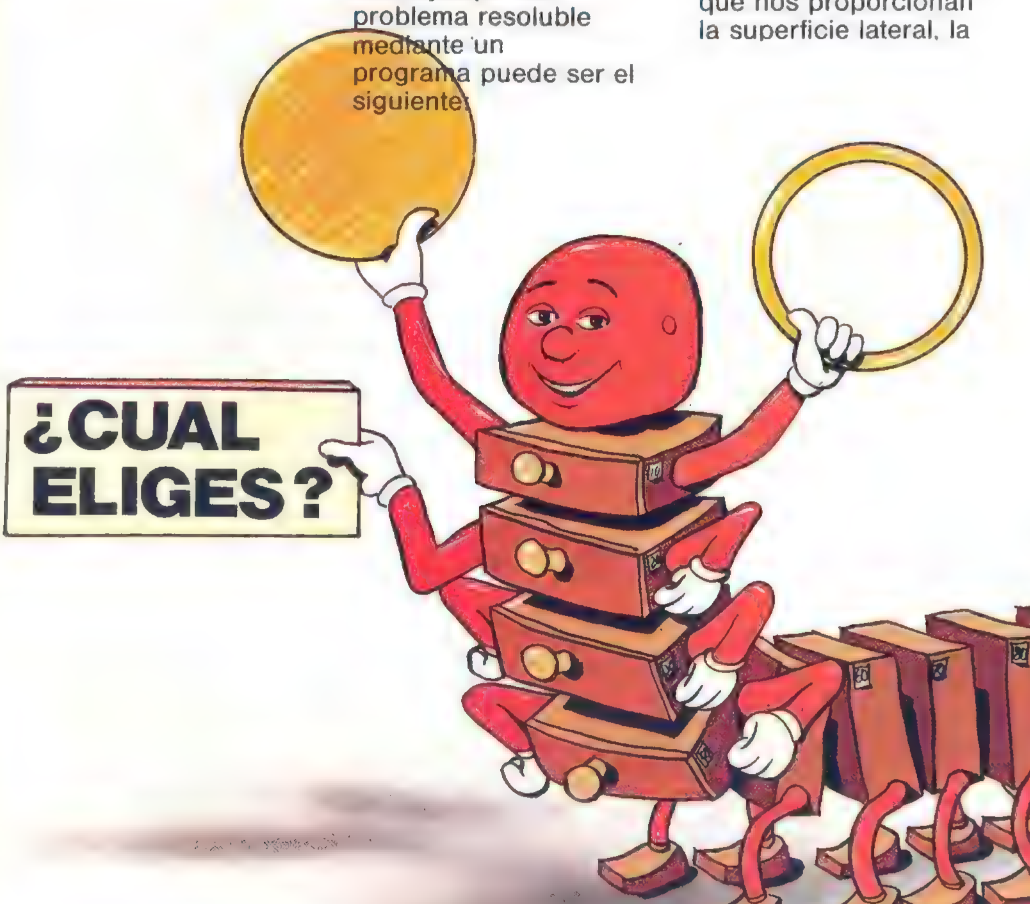
Otro ejemplo de problema resoluble mediante un programa puede ser el siguiente:

Objetivo

Queremos determinar la superficie lateral, la total y el volumen de un cilindro de cualquier dimensión.

Procedimiento

Para poder enfrentarnos con este problema, hay que conocer las fórmulas matemáticas que nos proporcionan la superficie lateral, la



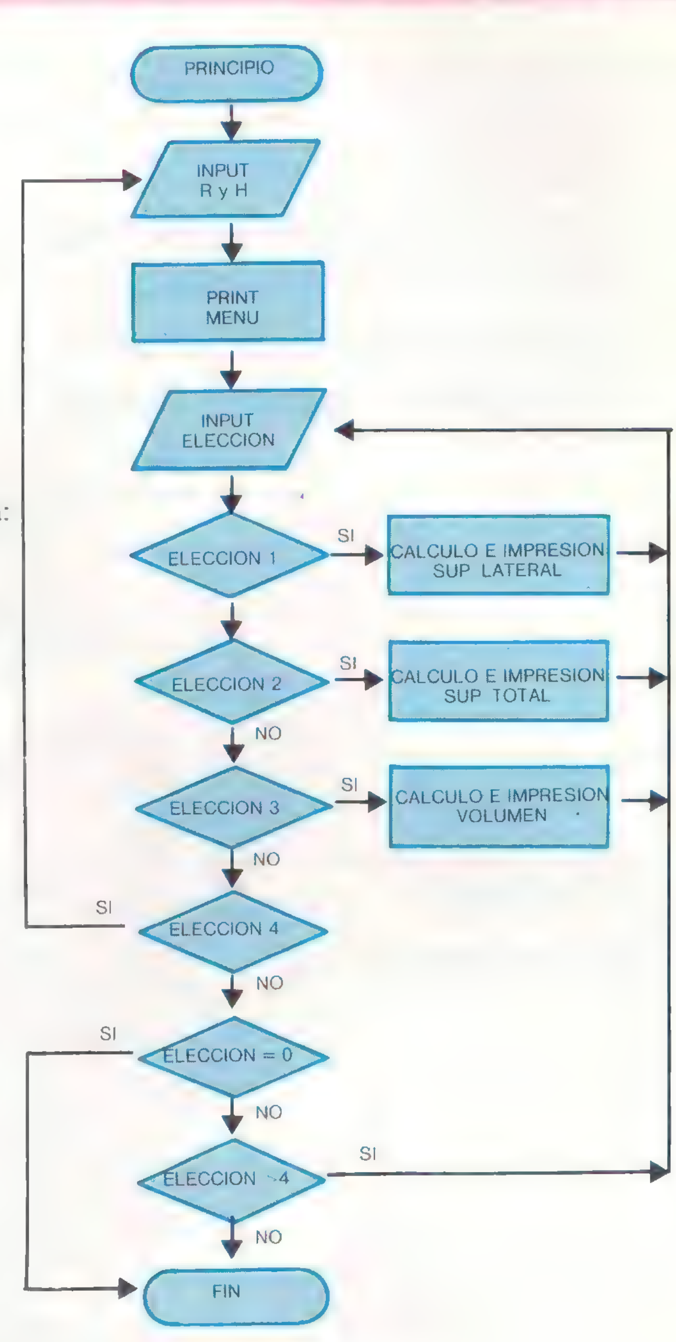
total, y el volumen de un cilindro. Veámoslas juntos.

a) La superficie lateral del cilindro (SLAT) se obtiene multiplicando la altura por la circunferencia de la base, recordando que esta última es el resultado de la siguiente operación: radio x 2 x 3,14. Expresión ésta, que en el lenguaje de tu Spectrum resulta: R * 2 * PI.

b) La superficie total del cilindro (SOT), se obtiene sumando las superficies de las dos bases circulares a la superficie lateral.
El área de un circulo se obtiene mediante la siguiente operación: radio x x radio x 3,14. Expresión que corresponde a R ↑ 2 * PI.

c) El volumen del cilindro (VOLUMEN) se obtiene multiplicando el área del circulo base por la altura = = R * R * PI * H.

Es bastante sencillo pasar al diagrama de flujo y de este al correspondiente listado BASIC.



PROGRAMACION

```
10 REM SUPERFICIE Y VOLUMEN DE UN CILINDRO
20 REM INTRODUCCION DATOS DEL CILINDRO
30 INPUT "RADIO = "; R
40 INPUT "ALTURA = "; H
50 REM IMPRIME EL MENU
60 PRINT "PULSA :"
70 PRINT "1 PARA SUPERFICIE LATERAL"
80 PRINT "2 PARA SUPERFICIE TOTAL"
90 PRINT "3 PARA VOLUMEN"
100 PRINT "4 PARA NUEVOS DATOS"
110 PRINT "0 PARA TERMINAR"
120 REM ELECCION
130 INPUT "ELIJO "; ELECCION
140 IF ELECCION = 1 THEN GOTO 200
150 IF ELECCION = 2 THEN GOTO 300
160 IF ELECCION = 3 THEN GOTO 400
170 IF ELECCION = 4 THEN GOTO 20
180 IF ELECCION = 0 THEN GOTO 500
190 IF ELECCION > 4 THEN GOTO 120
200 REM CALCULO E IMPRESION SUPERFICIE LATERAL
210 LET SLAT = R * 2 * PI * H
220 PRINT "SUPERFICIE LATERAL = "; SLAT
230 GOTO 120
300 REM CALCULO E IMPRESION SUPERFICIE TOTAL
310 LET SOT = R * 2 * PI * H + 2 * R * R * PI
320 PRINT "SUPERFICIE TOTAL = "; SOT
330 GOTO 120
400 REM CALCULO E IMPRESION VOLUMEN
410 LET VOLUMEN = R * R * PI * H
420 PRINT "VOLUMEN = "; VOLUMEN
430 GOTO 120
500 REM FIN
```

De estos dos simples ejemplos se desprenden ya algunas conclusiones fundamentales: un ordenador, gracias a su memoria y a los programas que tú le proporcionas, consigue desarrollar las tareas que le has asignado con mucha mayor precisión y, sobre todo, velocidad de como podria realizarlas cualquiera.
Un ordenador, además, puede elegir, por ejemplo, cuál entre dos números es el mayor, gracias al conjunto de conocimientos que tú le proporcionas.
Finalmente un ordenador es capaz de replantear un problema, tantas veces como sea necesario pudiendo escoger de nuevo los datos e informaciones que debe emplear.

EJERCICIOS

Anota en los espacios en blanco el resultado que preveas para **cada** ejercicio propuesto, y verificado después con las soluciones de tu **Spectrum**. Si has cometido un solo error, repasa la lección.

```
10 REM: LET N$ = "PEDRO"
20 PRINT N$
```

```
10 PRINT "NO CREO"
20 PRINT "QUE TU CONSIGAS"
30 CLS
40 PRINT "VER ALGO"
```

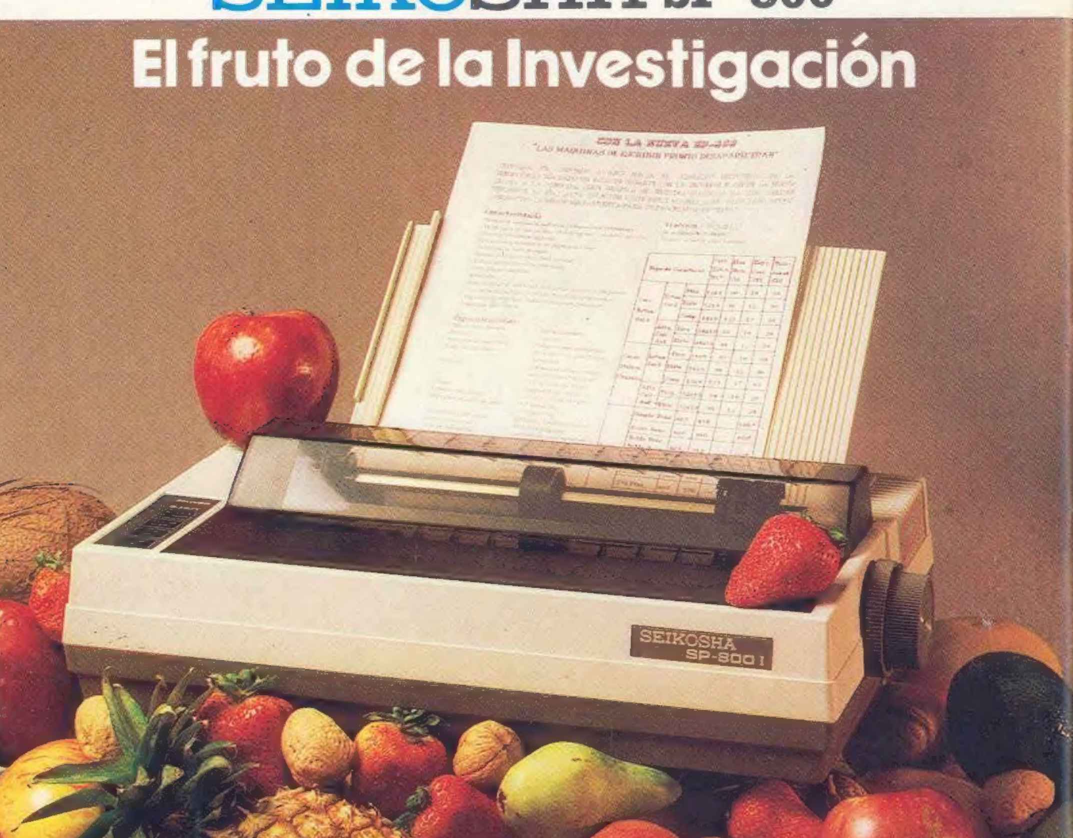
```
10 CLS
20 PRINT "AHORA SI"
```

```
10 CLS
20 REM: GOTO 40
30 PRINT "VIDEOBASIC"
40 PRINT "INGELEK"
```

```
10 LET P=0: LET S=1
20 IF P=1 THEN LET S=0
30 PRINT S, P
```

```
10 LET A = 129 : LET C = 131
20 IF A>C THEN LET C = A
30 IF C>A THEN LET A = C
40 PRINT A, C
```

SEIKOSHA SP-800



La nueva impresora de **SEIKOSHA SP-800**, con un ordenador personal puede escribir ***96 combinaciones de letra diferentes,*** desde 96 caracteres por segundo a 20 con muy alta calidad de letra, además es gráfica en alta densidad.

Su precio es de 69.900 Pt con ***introductor automático hoja a hoja.***

Con un pequeño ordenador personal, un procesador de textos puede costar alrededor de cien mil pesetas.

Infórmese y comprenderá por qué **las máquinas de escribir** tienen demasiados años.

Nuestra calidad es "SEIKO";
nuestros precios, únicos
Si desea más información,
consulte con nuestro distribuidor
más cercano, llame o escriba a:

DiRAO

DIRECCION COMERCIAL:
Av. Blasco Ibañez, 114-116
46022 VALENCIA
Tel (96) 372 80 09
Télex 62220
DIRECCION COMERCIAL EN CATALUÑA:
C/Muntaner, 60-2-4Pta
08011 BARCELONA
Tel (93) 323 32 19

ESTOS SON NUESTROS MODELOS:

MODELO	VELOCIDAD	COLUMNAS	TIPOS DE LETRA	P.V.P.R.* INTERFACE PARALELO
GP-50 LA PEQUERA	40 cps	46	2	25.900
GP-500 LA ECONOMICA	50 "	80	2	47.900
GP-550 LA STANDARD	86 "	80-136	10	59.900
SP-800 LA PERFECCION	96 "	80-137	20	69.900
GP-700 LA DE COLOR	50 "	80-106	3	84.900
BP-5200 LA DE OFICINA	200 "	136-272	10	199.900
BP-5420 LA MAS RAPIDA	420 "	136-272	10	299.900

* Los precios indicados son los recomendados para conexión tipo paralelo Centronics, para otro tipo de conexión, sufren un ligero incremento.

Este pie de página ha sido realizado integramente con la nueva impresora!

SEIKOSHA SP-800